PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-175698
(43)Date of publication of application : 14.07.1995

(51)Int.Cl. G06F 12/00

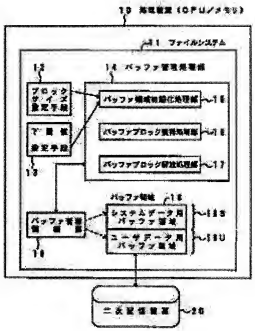
(21)Application number : 05-317832
(22)Date of filing : 17.12.1993

(71)Applicant : FUJITSU LTD
(72)Inventor : MURAKAMI TAKEO

(54) FILE SYSTEM

(57)Abstract:
PURPOSE: To improve input/output performance by dividing a buffer area for file input/output into areas by purposes including buffer areas for system data and user data, independently managing the respective areas, and controlling the possession and release of a requested buffer.

CONSTITUTION: Concerning the file system for a computer system provided with a central processing unit, main storage device and second-order storage device 20, a buffer area 18 for file input/output is initialized by a buffer area initialize processing part 15 and constituted so as to be divided into the plural areas by purposes such as a buffer area 18S for system data and a buffer area 18U for user data at this time of initializing. A buffer management processing part 14 independently manages the respective areas, and a buffer block possession processing part 16 and a buffer block release processing part 17 controls the possession and release of the requested buffer. Thus, input/output processing can be accelerated by utilizing the buffer for a parallel computer system or the like.



* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1]In a file system in a computer system provided with a central processing unit, a main memory unit, and a secondary memory, Divide the buffer space (18) into a buffer space for file input and output (18), and a field according to two or more uses which include a use of a buffer space for system data (18S), and a buffer space for user data (18U) at least, and each field independently. A file system provided with a buffer management treating part (14) which manages and controls acquisition and release of a buffer which were demanded.
[Claim 2]Have a block size setting means (12) which specifies size of a block which is an executive unit of a buffer in the file system according to claim 1 as every [which was divided according to said use] buffer space (18S, 18U), and Said buffer management treating part. A file system, wherein (14) is constituted so that acquisition and release of a buffer which managed a field and made the block size a unit with block size beforehand specified as said every divided buffer space (18S, 18U) may be controlled.
[Claim 3]In the file system according to claim 1, said buffer management treating part (14), A file system constituting so that it may change into a state which can assign a block in use and beyond a predetermined value may secure the number of assignable blocks, if the number of blocks which exist in a buffer space, and which can be assigned is less than a predetermined value.
[Claim 4]In the file system according to claim 1, a number of a block which can be assigned of lower limits which exist in each buffer space about a block which is an executive unit of a buffer to every [which was divided according to said use] buffer space (18S, 18U). Have a lower limit setting means (13) to specify, and said buffer management treating part (14), If the number of blocks which exist in a buffer space and which can be assigned is less than a block lower limit which was beforehand specified as said every divided buffer space (18S, 18U) and which can be assigned, A file system constituting so that it may change into a state which can assign a block in use and more than a block lower limit that can be assigned may secure the number of assignable blocks.
[Claim 5]In a file system in a computer system provided with a central processing unit, a main memory unit, and a secondary memory, A buffer block is acquired from a buffer space for file input and output (18), and said buffer space (18) to swap spaces. When it swaps out, set a dirty display which shows that the contents of the block have not been written in a secondary memory as information which manages the block, release the block to it, and to it for swapping in. A swap control means which cancels said dirty display and releases the block after swapping in when a buffer block is acquired, Said buffer space (18). By managing, and returning the contents of the block with which said dirty display is set up to a secondary memory, when the number of blocks which exist in a buffer space and which can be assigned is less than a block lower limit which was specified beforehand and which can be assigned, the number of assignable blocks. A file system provided with a buffer management treating part (14) which more than a block lower limit that can be assigned secures.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application]This invention relates to the file system which performs buffer management in consideration of input and output of system data, such as a swap.
[0002]In recent years, increase of the data volume treated in fields, such as large-scale numerical computation and time varying image processing, is remarkable, and the program itself has complicated and large-scale-ized it. Therefore, the computer system which can process more nearly mass data at high speed is called for. On the other hand, by new development of highly efficient CPU, maturation of parallel processing technology, etc., the improvement in computation capability is remarkable, and large scale-ization of a secondary storage and main memory is also following it. However, compared with these, the elongation of the data transfer rate of a secondary storage is small, and serves as a bottleneck of the whole system.
[0003]Then, main memory is used as a buffer of a secondary storage, and attaining improvement in the speed of radial transfer is usually performed. However, since there is a limit also in a main storage capacity, it is necessary to manage the limited buffer space efficiently.
[0004]
[Description of the Prior Art]In the conventional system, the whole buffer space was divided into the fixed-length block, and buffer management was performed. In a buffer space, the file which systems, such as a swapfile and dumping, use, and the file which users, such as a load module and a data file, use will be intermingled. Therefore, when the swap was performed or acquisition of dumping was performed, the buffer space might be occupied by the system, and a user program could not secure sufficient buffer space, but the problem that input-and-output performance fell might arise.
[0005]Since buffers with an enough system were not able to be secured if a swap is started when the job which outputs and inputs mass data runs and the buffer space is occupied, there was a problem that processing of a swap might become slow.
[0006]
[Problem(s) to be Solved by the Invention]As mentioned above, in the system which adopted the conventional buffer management system, Since one buffer space was used by the object for system data, and the object for user data, having competed, one of data occupied almost all buffer spaces, and there was a problem that the input-and-output performance of another side might fall extremely.
[0007]An object of this invention is to aim at improvement in input-and-output performance by dividing a buffer space according to a use and performing buffer management in consideration of the character of input and output.
[0008]
[Means for Solving the Problem]In the invention according to claim 1, as shown, for example in drawing 1, the buffer space 18 for file input and output is divided according to a use. A divided field

is called partition. One is the buffer space 18S for system data which systems secured when a job (process) is swapped out, such as a swapfile and dumping, use. Another is the buffer space 18U for user data which a user program uses. The buffer management treating part 14 manages each buffer space independently, and controls acquisition and release of a buffer which were demanded. The buffer space 18S for system data or the buffer space 18U for user data may be further divided according to a use.

[0009]When setting up a buffer at the time of starting of a system, it enables it to specify block size which is an executive unit of a buffer by the block size setting means 12 in the invention according to claim 2 as every [which was divided according to a use] buffer space (partition). The buffer management treating part 14 manages the buffer space 18S for system data, and the buffer space 18U for user data in block size different, respectively, and controls acquisition and release of a buffer which made the block size a unit by specified block size.

[0010]In the invention according to claim 3, the buffer management treating part 14 has managed the number of blocks which can be assigned for every partition. An assignable block is a block which is in agreement with the contents on a disk, although a block with which intact and effective data is not stored, or effective data is stored. Effective data is stored and the contents cannot assign a block (this is hereafter called dirty block) which is not reflected in a disk top.

[0011]When the buffer management treating part 14 performs block assignment, it investigates whether it is less than constant value (block lower limit which can be assigned) with the block count which can be assigned. If it has become less than constant value, the contents of the dirty block under list will be returned to the secondary memories 20, such as a disk, and the number of assignable blocks will be increased.

[0012]When setting up a buffer at the time of starting of a system, it enables it to specify a block lower limit which can be assigned by the lower limit setting means 13 in the invention according to claim 4 as every [which was divided according to a use] buffer space (partition). It is based on this specified block lower limit that can be assigned when the buffer management treating part 14 secures the block count in which assignment beyond constant value is possible.

[0013]In the invention according to claim 5, a context of an executing job swapped out or a process is written in a swapfile. At this time, if sufficient buffer space is secured to swapfiles, data will be written in a buffer, and a dirty flag which shows that a buffer block has not been written in to the secondary memory 20 is set. Even if a block with which a dirty flag was set is released, it assigns and it is impossible, but if required, it can be changed into a block which can be assigned by write return to the secondary memory 20 at any time. Then, if swapping in of a job is performed and a swapfile is read again, control which clears a dirty flag of an applicable buffer block will be performed.

[0014]

[Function]In the invention according to claim 1, since both buffer spaces 18S and 18U are divided even when the data input/output for systems and the data input/output for users are performed in parallel, it is guaranteed that all can secure a constant rate of buffer spaces. Therefore, each buffer spaces 18S and 18U for system data and for user data can be used, and it can output and input at a stable speed.

[0015]Since the character of input and output and the size of data differ between the data for systems, and the data for users greatly, they differ also in the executive unit of the optimal buffer for improving the utilization efficiency of a buffer, and input-and-output performance. In the invention according to claim 2, for every partition, specification of block size is possible and respectively suitable block size can be set up in consideration of the data of each partition, and input-output behavioral characteristics.

[0016]If a new buffer acquisition request comes when the assignable block count is lost in a buffer, after returning a dirty block to a disk, it will be necessary to assign, and the waiting time of buffer acquisition will increase. In the invention according to claim 3, since beyond constant value secures

the block which can always be assigned, there is no waiting time of buffer acquisition and input-and-output performance can be raised.
[0017]Since the size of input and output and an access pattern change with uses, the optimum values of the block lower limit which can be assigned differ for every partition. In the invention according to claim 4, since setting out of a different block lower limit for every partition which can be assigned can be performed, input-and-output performance can be raised by setting up a value suitable for the operating characteristic of each partition, respectively.
[0018]In swap control of an executing job or a process, once a swapfile is read by swapping in, it will not be read again. Therefore, when swapping in happens, even if the dirty block remains on the buffer, it is not necessary to return the contents to the secondary memories 20, such as a disk. In the invention according to claim 5, since a dirty flag is cleared using this character at the time of swapping in, write return on a useless disk can be prevented, and input-and-output performance can be raised. Since swapping out and swapping in are performed only on a buffer, without accessing to the secondary memory 20 when a buffer has a margin, high-speed swap control is possible.
[0019]
[Example]Drawing 1 shows the example of composition of this invention. 10 in drawing 1 A central processing unit (CPU). And a main memory unit. (Memory) etc. -- from -- the becoming processing unit and 11 the input and output to a file. The file system to perform, the block size setting means as which 12 specifies the block size of a buffer for every partition, the lower limit setting means which specifies a number of the block which can be assigned of lower limits with which 13 exists in a buffer space for every partition, and 14 a buffer. It is a buffer management treating part to manage, What consists of the buffer space initialization processing part 15, the buffer block acquisition treating part 16, the buffer block release treating part 17, etc., the buffer space where 18 was divided into the partition of the buffer space 18S for system data, and the buffer space 18U for user data, The buffer management information bureau where 19 holds the management information of the buffer space 18, and 20 express secondary memories, such as a disk unit.
[0020]The buffer space initialization processing part 15 is a processing means to initialize the buffer space 18 and to set the management information of the partition as the buffer management information bureau 19 according to the parameter specified for every partition by the block size setting means 12 and the lower limit setting means 13. The buffer block acquisition treating part 16 is a processing means to acquire a buffer block from an applicable partition to the acquisition request of the buffer from the demand origin which uses a buffer. The buffer block release treating part 17 is a processing means to release the block to the release request of the gained buffer.
[0021]As shown in drawing 1, the buffer space 18 for file input and output is initialized by the buffer space initialization processing part 15, and is constituted by the field according to use of plurality, such as the buffer space 18S for system data, and the buffer space 18U for user data, at the time of this initialization so that division is possible. The buffer management treating part 14 manages each field independently, and controls the acquisition and release of a buffer which were demanded by the buffer block acquisition treating part 16 and the buffer block release treating part 17.
[0022]Drawing 2 is an example explanatory view of composition of the file system in working example of this invention. In drawing 2, the thing of drawing 1 and a same sign to what is shown in drawing 1. Corresponding, program A-C, such as an application program with which 30A-30C perform the input and output to a file, the file management processing part which 31 receive the input/output request to a file and is processed, and 32 express the disk unit in which a file is stored.

[0023]The file system 11 comprises the buffer space 18 the file management processing part 31, the buffer management treating part 14, and for file input and output, etc. The file management processing part 31 is a treatment module which receives input/output request from an application program and performs a data area peculiar to a program, the buffer of the file system 11, and data input/output processing between the disk units 32.

[0024]The buffer management treating part 14 is a treatment module which manages the buffer space 18 used with the file system 11, and performs acquisition of a buffer, and release processing by a block unit by the demand from the file management processing part 31. The buffer management treating part 14 is divided into the buffer space initialization processing part 15, the buffer block acquisition treating part 16, and the buffer block release treating part 17.
[0025]According to a use, it divides and the buffer space 18 is managed, as shown in drawing 3 (A). In this example, the buffer space 18 is divided into three partitions, the partition 1 is assigned as a buffer space for system data, and the two partitions 2 and 3 are assigned as a buffer space for user data.
[0026]The block used as the executive unit of acquisition with each partition and release can specify the size beforehand for every partition. Since input and output with much data volume generally increase like a swap or dumping in the case of system data, in this example, large block size is set up as size of the buffer blocks B11-B13 in the partition 1 for system data. On the other hand, in the partition 2 for user data, since there are generally much input and output of a small unit, it is set up so that the size of the buffer blocks B21-B24 may become small.
[0027]The lower limit of the number of blocks which can be beforehand assigned at the time of initialization can be specified now to each partition. These information is managed for every partition as partition management information, as shown in drawing 3 (B). This partition management information is held in the buffer management information bureau 19 which shows drawing 1.
[0028]Drawing 4 is a management information explanatory view of the buffer block in working example of this invention. Hash cue and a free list are held for every partition as a data structure for managing the buffer space 18 other than partition management information to a block unit. 40 shown in drawing 4 by the block which can be assigned, and disk writing return. A hash table for the free list for managing the block which can be changed into the block which can be assigned, and 41 to manage hash cue, and 42 express the buffer block information which is the management information provided in each block correspondence.
[0029]As shown, for example in drawing 4 (A) for management of a buffer block, the hash table 41 which has the hash value 0 - a cue pointer corresponding to H for every partition is formed, and the buffer block information 42 into which effective data went for every hash value is held in the form of a list. A hash value is at this example. [File ID and the position in a file] are used as the key. File ID is file identification information given to a meaning within a system. The position in a file is the offset (displacement) information from the head in the file.
[0030]The free list 40 holds the buffer block information 42 used as the object seized when assigning a new buffer block in the form of a list. The buffer block information 42 established corresponding to each buffer block has the field where the information on the position in a file is set to file ID corresponding to the contents of the block, as shown, for example in drawing 4 (B). It has the field of the addresses in a buffer of the block (for example, offset or a block number from a partition head, etc.). It has a data valid flag which shows that the contents of the dirty flag which shows not having written the contents of the block in a disk to others, and its block are effective. It has a pointer for the chaining of hash cue, a pointer for the chaining of a free list, etc.
[0031]Next, the swap control using the buffer management system by this invention is explained. Drawing 5 is a swap control explanatory view in working example of this invention.
[0032]A swap is processing which vacates the storage area which evacuated to other fields and occupied till then the storage area which the job is using now for other jobs. In the case of this example, as shown in drawing 5 (A), the context 50 of a job is swapped out to the block B11 acquired to the buffer space 18, and the dirty flag of the block B11 is made one. And the buffer block information 42 of the block B11 is connected to the free list 40. At the time of swapping in, the block B11 is acquired by the acquisition request of a buffer block, and the contents are restored as the context 50 of a job. If swapping in is carried out, the dirty flag of the block B11 will be cleared.

[0033]When the number of the blocks of the buffer space 18 which can be assigned becomes less than a predetermined lower limit, the contents of the block B11 under swap are returned to the disk unit 32, and are treated as a block which can be assigned. At the time of swapping in, when the data of the block B11 is not effective, the new block B12 is acquired, the swapped context 50 is read from the disk unit 32, and the context 50 is restored to a new job field from the block B12.
[0034]The swapfile 51 will be realized by the buffer space 18 and the field of the disk unit 32 by making it above. Since a dirty flag is cleared when swapping in is performed, the write return to the useless disk unit 32 can be prevented. Namely, when there is sufficient margin for the buffer space 18, access to the disk unit 32 becomes unnecessary in the case of swapping out/swapping in.
[0035]The block with which the dirty flag is turned on is called dirty block. A dirty block is a block which is not reflecting the contents on a disk. In the free list 40 shown in drawing 5 (B), the buffer block information 42b and 42d and the block with —42x are dirty blocks. Other blocks are blocks which can be assigned, if the number of these blocks that can be assigned becomes smaller than the block lower limit which was set up for every partition and which can be assigned, the writing to the disk unit 32 of a dirty block will be performed, and a dirty flag will be cleared. Thereby, as shown in drawing 5 (C), the number of the blocks which can be assigned is managed so that it may become more than [which can be assigned] a block lower limit.
[0036]Drawing 6 is a figure showing the change state of the block under swap in working example of this invention. In the state which shows in drawing 6 (A), if a buffer block is acquired for swapping out, required number reservation is carried out, and after swapping out, the block for swapfiles will make a dirty flag one, and will be released. The state of the buffer block information 42 connected with the free list 40 comes to be shown in drawing 6 (B). If swapping in is performed, the dirty flag of the block used as a swapfile will be returned at OFF, and it will change in the state where it is shown in drawing 6 (C).
[0037]Drawing 7 is a buffer space initialization processing flow chart by working example of this invention. The buffer space initialization processing part 15 shown in drawing 2 is called from the file management processing part 31 at the time of initialization of a system. The input parameters of an argument are the information on a buffer space, block size, and a block lower limit that can be assigned (Step 70 of drawing 7). Step 71 divides the buffer space initialization processing part 15 first in the block size unit which had the object domain specified. Next, by Step 72, about each block, the buffer block information 42 of the contents shown in drawing 4 (B) is initialized, and it puts into the free list 40. In Step 73, the specified block lower limit which can be assigned is set up as partition management information. In Step 74, the number of whole blocks is initialized as the block count which can be assigned. In Step 75, partition ID is assigned and the partition ID is returned in the file management processing part 31 of a calling agency (Step 76).
[0038]By calling the buffer space initialization processing part 15 according to the use of a buffer space, the file management processing part 31 can divide into plurality the buffer space 18 used with the file system 11, and can set up a partition according to a use.
[0039]Drawing 8 is a buffer block acquisition processing flow chart by working example of this invention. The buffer block acquisition treating part 16 shown in drawing 2 is called from the file management processing part 31 at the time of the data input/output of each programs 30A–30B. Input parameters are partition ID which specifies the partition which is the acquisition target of a block, file ID for a data input/output, and the information on the position in a file (Step 800 of drawing 8).
[0040]The buffer block acquisition treating part 16 calculates the hash value which used specified file ID and the position in a file as the key by Step 801 first. Next, based on the calculated hash value, by Step 802, it asks for the entry of the hash table 41 in the applicable partition shown in drawing 4, and looks for hash cue.
[0041]If the applicable buffer block information 42 whose file ID and position in a file correspond is found with the judgment of Step 803, when it progresses to Step 810 and is not found, it progresses

to the following step 804.
[0042]When the applicable buffer block information 42 is not found, by Step 804, it searches for the free list 40 and the buffer block information 42 of the block which was found first and which can be assigned is removed from the free list 40. In Step 805, the buffer block information 42 of this block is removed from old hash cue, and it puts in to the new hash cue corresponding to the hash value which calculated file ID and the position in a file as a key. At the following step 806, the buffer block information 42, such as setting out of file ID, setting out of the position in a file, and OFF of a data valid flag, is initialized, and the block count which can be assigned is reduced at Step 807.
[0043]By Step 808, it judges whether the block count which can be assigned became smaller than the block lower limit which can be assigned, and if not small, it progresses to Step 813. If smaller than the block lower limit which can be assigned, by Step 809, it searches for the free list 40, and the contents of the block with which the dirty flag is turned on will be returned to a disk, the block will be changed into the state which can be assigned, and the block count which can be assigned will be increased. Then, it progresses to Step 813.
[0044]By the judgment of Step 803, when the applicable buffer block information 42 is found, it is judged by Step 810 whether the buffer block information 42 is among the free list 40. If it is among the free list 40, the buffer block information 42 will be removed from the free list 40 by Step 811.
[0045]Next, by Step 812, the data valid flag in the buffer block information 42 is made one, and it progresses to Step 813. In Step 813, it returns to the file management processing part 31 of a calling agency by making buffer block information 42 of the acquired block into a print-out.
[0046]Drawing 9 is a buffer block release processing flow chart by working example of this invention. The buffer block release treating part 17 shown in drawing 2 is called from the file management processing part 31, and performs release processing of the acquired buffer block. Input parameters are partition ID which specifies the partition with which the block to release exists, buffer block information, and a swapping in flag (Step 90 of drawing 9). Swapping in flags are one and a flag which is set up at OFF in the case of others, when releasing the buffer block with which swapping in was performed.
[0047]The buffer block release treating part 17 puts the buffer block information 42 of the block to release into the free list 40 of an applicable partition by Step 91 first. Next, by Step 92, a swapping in flag judges one or OFF, and when a swapping in flag is one (i.e., when it is block release at the time of swapping in), a dirty flag is cleared by Step 93.
[0048]In Step 94, ON and OFF of a dirty flag is judged, and if the dirty flag is off, the block count which can be assigned will be increased. Then, it returns to the file management processing part 31 of a calling agency at Step 96.
[0049]Drawing 10 is a swap processing flow chart by working example of this invention. Swap control of the job by the swapping in processing shown in the file management processing part 31 or the swapping out processing shown in drawing 10 (A) by the processing means of the swap control which carried out figures omitted abbreviated, and drawing 10 (B) which are shown in drawing 2 is performed.
[0050]In swapping out, first, by Step 101, partition ID of the buffer space 18S for system data, file ID of a swapfile, and the position in a file are specified, the buffer block acquisition treating part 16 is called, and the block for swapfiles is acquired. Next, the context 50 of the job swapped out to the acquired buffer block by Step 102 is written in, and the dirty flag in the buffer block information 42 of the block is made one by Step 103.
[0051]Next, a swapping in flag is cleared by Step 104, and the buffer block release treating part 17 is called by Step 105 by making partition ID, buffer block information, and a swapping in flag into an argument. By Step 106, if control returns from the buffer block release treating part 17, if it judges whether swapping out of all the contexts 50 was completed and there is an unsettled portion, it will return to Step 101 and processing will be similarly repeated about the following block. If it ends about all the contexts 50, processing of swapping out will be ended.

[0052]In swapping in, first, by Step 111, partition ID of the buffer space 18S for system data, file ID of a swapfile, and the position in a file for swapping in are specified, and the buffer block acquisition treating part 16 is called. If a buffer block is acquired, with reference to the buffer block information 42, ON and OFF of a data valid flag will be judged by Step 112. If a data valid flag is one, it will progress to Step 114.
[0053]If the data valid flag is off, the applicable context 50 will be read from the swapfile on a disk to the acquired block by Step 113. In Step 114, the contents of the buffer block are copied to a job field, and the context 50 is restored. Then, a swapping in flag is made one by Step 115, and the buffer block release treating part 17 is called by Step 116 by making partition ID, buffer block information, and a swapping in flag into an argument. By Step 117, if control returns from the buffer block release treating part 17, if it judges whether the swapping in of all the contexts 50 was completed and there is an unsettled portion, it will return to Step 111 and processing will be similarly repeated about the following block. If it ends about all the contexts 50, processing of swapping in will be ended.
[0054]
[Effect of the Invention]As explained above, according to this invention, it becomes possible to perform buffer management according to use which suited the use of each input and output, and largely contributes to the improved efficiency of the whole system. In the parallel computer system etc. which comprise a processing unit of a large number which it says especially are tens of sets – thousands of sets, for example, The art of managing efficiently the buffer space limited in each processing unit, and using it is needed, and according to this invention, it becomes possible to realize improvement in the speed of radial transfer which used the buffer in such a system, and effective use of a buffer space.

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号
特開平7－175698
(43)公開日　平成７年(1995)７月14日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/00	５１４　Ａ	8944－5B		

審査請求　未請求　請求項の数５　ＯＬ　（全 10 頁）

(21)出願番号　特願平5－317832

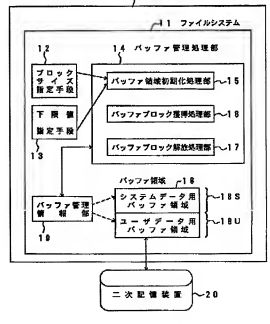
(22)出願日　平成５年(1993)12月17日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地
(72)発明者　村上　岳生
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内
(74)代理人　弁理士　小笠原　吉義　（外２名）

(54)【発明の名称】　ファイルシステム

(57)【要約】
【目的】スワップ等のシステムデータの入出力を考慮してバッファ管理を行うファイルシステムに関し，入出力の各用途に適したバッファ管理により，入出力性能の向上を図ることを目的とする。
【構成】ファイル入出力用のバッファ領域18を，システムデータ用，ユーザデータ用というような用途別に複数の領域に分割し，バッファ管理処理部14により，それぞれの領域を独立に管理し，要求されたバッファの獲得および解放を制御する。また，用途別に分割したバッファ領域18S，18U 毎に，ブロックサイズおよび割り当て可能ブロック下限値を指定する手段を設け，領域の効率的利用とバッファ獲得の待ち時間の短縮を可能とする。

本発明の構成例



1

【特許請求の範囲】

【請求項1】　中央処理装置，主記憶装置および二次記憶装置を備えた計算機システムにおけるファイルシステムにおいて，ファイル入出力用のバッファ領域(18)と，そのバッファ領域(18)を，少なくともシステムデータ用バッファ領域(18S)およびユーザデータ用バッファ領域(18U)の用途を含む複数の用途別の領域に分割し，それぞれの領域を独立して管理し，要求されたバッファの獲得および解放を制御するバッファ管理処理部(14)とを備えたことを特徴とするファイルシステム。

【請求項2】　請求項1記載のファイルシステムにおいて，前記用途別に分割したバッファ領域(18S,18U)毎に，バッファの管理単位であるブロックのサイズを指定するブロックサイズ指定手段(12)を備え，前記バッファ管理処理部(14)は，前記分割したバッファ領域(18S,18U)毎にあらかじめ指定されたブロックサイズによって領域を管理し，そのブロックサイズを単位としたバッファの獲得および解放を制御するように構成されたことを特徴とするファイルシステム。

【請求項3】　請求項1記載のファイルシステムにおいて，前記バッファ管理処理部(14)は，バッファ領域内に存在する割り当て可能なブロックの数が所定値を下回ると，使用中のブロックを割り当て可能な状態に変更し，割り当て可能なブロックの数を所定値以上確保するように構成されたことを特徴とするファイルシステム。

【請求項4】　請求項1記載のファイルシステムにおいて，前記用途別に分割したバッファ領域(18S,18U)毎に，バッファの管理単位であるブロックについて，それぞれのバッファ領域内に存在する割り当て可能なブロックの数の下限値を指定する下限値指定手段(13)を備え，前記バッファ管理処理部(14)は，バッファ領域内に存在する割り当て可能なブロックの数が前記分割したバッファ領域(18S,18U)毎にあらかじめ指定された割り当て可能ブロック下限値を下回ると，使用中のブロックを割り当て可能な状態に変更し，割り当て可能なブロックの数を割り当て可能ブロック下限値以上確保するように構成されたことを特徴とするファイルシステム。

【請求項5】　中央処理装置，主記憶装置および二次記憶装置を備えた計算機システムにおけるファイルシステムにおいて，ファイル入出力用のバッファ領域(18)と，前記バッファ領域(18)からバッファブロックをスワップ領域用に獲得してスワップアウトをした際に，そのブロックを管理する情報に，ブロックの内容が二次記憶装置に未書き込みであることを示すダーティ表示を設定してそのブロックを解放し，スワップインのためにバッファブロックを獲得した際には，スワップイン後に前記ダーティ表示を解除してそのブロックを解放するスワップ制御手段と，前記バッファ領域(18)を管理し，バッファ領域内に存在する割り当て可能なブロックの数があらかじめ指定された割り当て可能ブロック下限値を下回った場合に，前記ダーティ表示が設定されているブロックの内容を二次記憶装置に書き戻すことにより，割り当て可能なブロックの数を割り当て可能ブロック下限値以上確保するバッファ管理処理部(14)とを備えたことを特徴とするファイルシステム。

2

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は，スワップ等のシステムデータの入出力を考慮してバッファ管理を行うファイルシステムに関する。

【0002】近年，大規模数値計算や動画像処理等の分野において扱うデータ量の増大は著しく，またプログラム自体も複雑化し大容量化している。従って，より大容量のデータを高速に処理することが可能な計算機システムが求められている。一方，高性能なＣＰＵの新規開発，並列処理技術の成熟等により，計算処理能力の向上は著しく，また二次記憶，主記憶の大容量化も進んでいる。しかし，これらに比べて二次記憶のデータ転送速度の伸びは小さく，システム全体のボトルネックとなっている。

【0003】そこで通常，主記憶を二次記憶のバッファとして利用し，入出力処理の高速化を図ることが行われている。しかし，主記憶容量にも限界があるので，限られたバッファ領域を効率良く管理する必要がある。

【0004】

【従来の技術】従来のシステムにおいては，バッファ領域全体を固定長のブロックに分割してバッファ管理を行っていた。バッファ領域はスワップファイル，ダンプ等のシステムが使用するファイルと，ロードモジュール，データファイル等のユーザが使用するファイルとが混在することになる。従って，スワップが行われたり，ダンプの取得が行われたりすると，バッファ領域がシステムに占有され，ユーザプログラムが十分なバッファ領域を確保できず，入出力性能が低下するという問題が生じることがあった。

【0005】また，大容量のデータの入出力を行うジョブが走行し，バッファ領域を占有している際に，スワップを起動すると，システムが十分なバッファを確保できないため，スワップの処理が遅くなることがあるという問題があった。

【0006】

【発明が解決しようとする課題】以上のように，従来のバッファ管理方式を採用したシステムにおいては，１つのバッファ領域をシステムデータ用とユーザデータ用とで競合して使用しているので，いずれかのデータがほとんどのバッファ領域を占有してしまい，他方の入出力性能が極端に落ちてしまうことがあるという問題があった。

【0007】本発明は，用途別にバッファ領域を分割し，入出力の性質を考慮したバッファ管理を行うことに

より，入出力性能の向上を図ることを目的とする。

【０００８】

【課題を解決するための手段】請求項１記載の発明では，例えば図１に示すように，ファイル入出力用のバッファ領域１８を用途別に分ける。分割された領域をパーティションという。１つはジョブ（プロセス）がスワップアウトされる際に確保するスワップファイルやダンプ等のシステムが使用するシステムデータ用バッファ領域１８Ｓである。もう１つはユーザプログラムが使用するユーザデータ用バッファ領域１８Ｕである。バッファ管理処理部１４は，それぞれのバッファ領域を独立して管理し，要求されたバッファの獲得および解放を制御する。なお，システムデータ用バッファ領域１８Ｓまたはユーザデータ用バッファ領域１８Ｕをさらに用途別に分割してもよい。

【０００９】請求項２記載の発明では，システムの起動時にバッファの設定を行う際に，ブロックサイズ指定手段１２によって，用途別に分割したバッファ領域（パーティション）毎に，バッファの管理単位であるブロックサイズを指定できるようにする。バッファ管理処理部１４は，指定されたブロックサイズによってシステムデータ用バッファ領域１８Ｓおよびユーザデータ用バッファ領域１８Ｕを，それぞれ異なるブロックサイズで管理し，そのブロックサイズを単位としたバッファの獲得および解放を制御する。

【００１０】請求項３記載の発明では，バッファ管理処理部１４は，パーティション毎に割り当て可能なブロックの数を管理している。割り当て可能なブロックとは，未使用で有効なデータが格納されていないブロック，あるいは有効なデータが格納されているがディスク上の内容と一致しているブロックである。有効なデータが格納されており，その内容がディスク上に未反映のブロック（以下，これをダーティブロックという）は割り当て不可能である。

【００１１】バッファ管理処理部１４は，ブロック割り当てを行う際に，割り当て可能ブロック数がある一定値（割り当て可能ブロック下限値）より少ないかどうかを調べる。一定値より少なくなっていればリスト中のダーティブロックの内容をディスク等の二次記憶装置２０に書き戻し，割り当て可能なブロックの数を増やす。

【００１２】請求項４記載の発明では，システムの起動時にバッファの設定を行う際に，下限値指定手段１３により，用途別に分割したバッファ領域（パーティション）毎に，割り当て可能ブロック下限値を指定できるようにする。バッファ管理処理部１４が一定値以上の割り当て可能なブロック数を確保する際には，この指定された割り当て可能ブロック下限値を基準とする。

【００１３】請求項５記載の発明では，スワップアウトされる実行中ジョブまたはプロセスのコンテキストがスワップファイルに書き込まれる。このとき，十分なバッファ領域がスワップファイル用に確保されればデータをバッファに書き込み，バッファブロックが二次記憶装置２０に対して未書き込みであることを示すダーティフラグをセットする。ダーティフラグがセットされたブロックは解放されても割り当て不可能であるが，必要であれば二次記憶装置２０への書き戻しにより，いつでも割り当て可能ブロックに変更することができる。その後，ジョブのスワップインが行われ再びスワップファイルが読み込まれると，該当バッファブロックのダーティフラグをオフにする制御を行う。

【００１４】

【作用】請求項１記載の発明では，システム用のデータ入出力とユーザ用のデータ入出力とが並行して行われる場合でも，両者のバッファ領域１８Ｓ，１８Ｕが分割されているため，いずれも一定量のバッファ領域を確保できることが保証される。従って，システムデータ用，ユーザデータ用の各バッファ領域１８Ｓ，１８Ｕを使用して，安定した速度で入出力を行うことができる。

【００１５】システム用のデータとユーザ用のデータとは，入出力の性質やデータのサイズが大きく異なるので，バッファの利用効率，入出力性能を上げるための最適なバッファの管理単位も異なる。請求項２記載の発明では，パーティション毎にブロックサイズの指定が可能であり，各パーティションのデータ，入出力特性を考慮してそれぞれ適切なブロックサイズを設定することができる。

【００１６】割り当て可能なブロック数がバッファ中になくなった際に，新たなバッファ獲得要求が来ると，ダーティなブロックをディスクに書き戻してから割り当てる必要があり，バッファ獲得の待ち時間が増えてしまう。請求項３記載の発明では，常に割り当て可能なブロックを一定値以上確保しておくので，バッファ獲得の待ち時間がなく，入出力性能を向上させることができる。

【００１７】また，用途によって入出力のサイズ，アクセスパターンは異なるため，パーティション毎に割り当て可能ブロック下限値の最適値は異なる。請求項４記載の発明では，パーティション毎に異なる割り当て可能ブロック下限値の設定ができるため，各パーティションの使用特性に適した値をそれぞれ設定することにより，入出力性能を向上させることができる。

【００１８】実行中ジョブまたはプロセスのスワップ制御において，スワップファイルが一度スワップインで読み込まれると，再び読み込まれることはない。従って，スワップインが起こった時点でダーティなブロックがバッファ上に残っていても，その内容をディスク等の二次記憶装置２０へ書き戻す必要はない。請求項５記載の発明では，この性質を利用してスワップイン時にダーティフラグをオフにするので無駄なディスクへの書き戻しを防ぐことができ，入出力性能を向上させることができる。また，バッファに余裕がある場合には，二次記憶装

置２０へアクセスすることなく，バッファ上だけでスワップアウト，スワップインが行われるので，高速なスワップ制御が可能である。

【００１９】

【実施例】図１は本発明の構成例を示す。図１において，１０は中央処理装置（ＣＰＵ）および主記憶装置（メモリ）等からなる処理装置，１１はファイルへの入出力を行うファイルシステム，１２はパーティション毎にバッファのブロックサイズを指定するブロックサイズ指定手段，１３はパーティション毎にバッファ領域内に存在する割り当て可能なブロックの数の下限値を指定する下限値指定手段，１４はバッファを管理するバッファ管理処理部であって，バッファ領域初期化処理部１５，バッファブロック獲得処理部１６，バッファブロック解放処理部１７などからなるもの，１８はシステムデータ用バッファ領域１８Ｓとユーザデータ用バッファ領域１８Ｕのパーティションに分割されたバッファ領域，１９はバッファ領域１８の管理情報を保持するバッファ管理情報部，２０はディスク装置等の二次記憶装置を表す。

【００２０】バッファ領域初期化処理部１５は，パーティション毎にブロックサイズ指定手段１２および下限値指定手段１３で指定されたパラメータに従って，バッファ領域１８の初期化を行い，そのパーティションの管理情報をバッファ管理情報部１９に設定する処理手段である。バッファブロック獲得処理部１６は，バッファを使用する要求元からのバッファの獲得要求に対して，該当するパーティションからバッファブロックを獲得する処理手段である。バッファブロック解放処理部１７は，獲得されたバッファの解放要求に対して，そのブロックを解放する処理手段である。

【００２１】図１に示すように，ファイル入出力用のバッファ領域１８は，バッファ領域初期化処理部１５によって初期化され，この初期化時にシステムデータ用バッファ領域１８ＳおよびユーザデータＵ用バッファ領域１８Ｕ等の複数の用途別の領域に分割可能に構成されている。バッファ管理処理部１４は，それぞれの領域を独立して管理し，バッファブロック獲得処理部１６およびバッファブロック解放処理部１７によって，要求されたバッファの獲得および解放を制御する。

【００２２】図２は，本発明の実施例におけるファイルシステムの構成例説明図である。図２において，図１と同符号のものは図１に示すものに対応し，３０Ａ～３０Ｃはファイルへの入出力を行うアプリケーションプログラム等のプログラムＡ～Ｃ，３１はファイルへの入出力要求を受け取って処理するファイル管理処理部，３２はファイルが格納されるディスク装置を表す。

【００２３】ファイルシステム１１は，ファイル管理処理部３１とバッファ管理処理部１４とファイル入出力用のバッファ領域１８などから構成されている。ファイル管理処理部３１は，アプリケーションプログラムから入出力要求を受け取り，プログラム固有のデータ領域，ファイルシステム１１のバッファ，ディスク装置３２の間のデータ入出力処理を行う処理モジュールである。

【００２４】バッファ管理処理部１４は，ファイルシステム１１で使用するバッファ領域１８を管理し，ファイル管理処理部３１からの要求により，ブロック単位でバッファの獲得，解放処理を行う処理モジュールである。バッファ管理処理部１４は，バッファ領域初期化処理部１５と，バッファブロック獲得処理部１６と，バッファブロック解放処理部１７とに分かれている。

【００２５】バッファ領域１８は，図３（Ａ）に示すように，用途別に分割して管理される。この例では，バッファ領域１８は３つのパーティションに分割され，システムデータ用のバッファ領域としてパーティション１が，ユーザデータ用のバッファ領域として２つのパーティション２，３が割り当てられている。

【００２６】各パーティションでの獲得，解放の管理単位となるブロックは，パーティション毎にあらかじめそのサイズを指定することができる。システムデータの場合，スワップやダンプ等のように一般にデータ量の多い入出力が多くなることから，この例では，システムデータ用のパーティション１におけるバッファブロックＢ１～Ｂ１３のサイズとして，大きいブロックサイズが設定されている。一方，ユーザデータ用のパーティション２では，一般に小さい単位の入出力が多いことから，バッファブロックＢ２１～Ｂ２４のサイズが小さくなるように設定されている。

【００２７】また，各パーティションに対して，初期化時にあらかじめ割り当て可能なブロックの数の下限値を指定できるようになっている。これらの情報は，図３（Ｂ）に示すように，パーティション管理情報として各パーティション毎に管理される。このパーティション管理情報は，図１に示すバッファ管理情報部１９に保持される。

【００２８】図４は本発明の実施例におけるバッファブロックの管理情報説明図である。パーティション管理情報の他に，バッファ領域１８をブロック単位で管理するためのデータ構造として，各パーティション毎にハッシュキュー，フリーリストを保持している。図４に示す４０は割り当て可能ブロックおよびディスク書き戻しによって割り当て可能ブロックに変更できるブロックを管理するためのフリーリスト，４１はハッシュキューを管理するためのハッシュテーブル，４２は各ブロック対応に設けられる管理情報であるバッファブロック情報を表す。

【００２９】バッファブロックの管理のために，例えば図４（Ａ）に示すように，各パーティション毎にハッシュ値０～Ｈに対応したキューポインタを持つハッシュテーブル４１が設けられ，ハッシュ値毎に有効なデータの入ったバッファブロック情報４２をリストの形で保持す

る。ハッシュ値は、この例では〔ファイルＩＤ，ファイル内位置〕をキーとしている。ファイルＩＤはシステム内で一意に付与されるファイル識別情報である。ファイル内位置は、そのファイルにおける先頭からのオフセット（変位）情報である。

【００３０】フリーリスト４０は，新規のバッファブロックを割り当てる際に横取りする対象となるバッファブロック情報４２をリストの形で保持している。個々のバッファブロックに対応して設けられるバッファブロック情報４２は，例えば図４（Ｂ）に示すように，そのブロックの内容に対応するファイルＩＤと，ファイル内位置の情報が設定されるフィールドを持つ。また，そのブロックのバッファ内アドレス（例えば，パーティション先頭からのオフセットまたはブロック番号等）のフィールドを持つ。他に，ブロックの内容がディスクに未書き込みであることを示すダーティフラグ，そのブロックの内容が有効であることを示すデータ有効フラグを持つ。さらに，ハッシュキーのチェイニングのためのポインタおよびフリーリストのチェイニングのためのポインタ等を持つ。

【００３１】次に，本発明によるバッファ管理方式を用いたスワップ制御について説明する。図５は，本発明の実施例におけるスワップ制御説明図である。

【００３２】スワップは，ジョブが現在使用している記憶領域を他の領域に退避して，それまで占めていた記憶領域を他のジョブ等のために明け渡す処理である。本実施例の場合，図５（Ａ）に示すように，ジョブのコンテキスト５０をバッファ領域１８に獲得したブロックＢ１１へスワップアウトし，ブロックＢ１１のダーティフラグをオンにする。そして，そのブロックＢ１１のバッファブロック情報４２をフリーリスト４０に接続しておく。スワップインのときには，バッファブロックの獲得要求によりブロックＢ１１を獲得し，その内容をジョブのコンテキスト５０として復元する。スワップインをしたならば，そのブロックＢ１１のダーティフラグをオフにする。

【００３３】スワップ中のブロックＢ１１の内容は，バッファ領域１８の割り当て可能ブロックの数が所定の下限値より少なくなった場合には，ディスク装置３２に書き戻され，割り当て可能ブロックとして扱われる。スワップインのとき，ブロックＢ１１のデータが有効でない場合には，新しいブロックＢ１２を獲得して，スワップされたコンテキスト５０をディスク装置３２から読み出し，そのブロックＢ１２から新たなジョブ領域にコンテキスト５０を復元する。

【００３４】以上のようにすることにより，スワップファイル５１は，バッファ領域１８とディスク装置３２の領域とによって実現されることになる。スワップインが行われたときには，ダーティフラグをオフにするので，無駄なディスク装置３２への書き戻しを防ぐことができる。すなわち，バッファ領域１８に十分な余裕がある場合には，スワップアウト／スワップインの際にディスク装置３２へのアクセスは不要となる。

【００３５】ダーティフラグがオンになっているブロックをダーティブロックという。ダーティブロックは，ディスク上に内容を反映していないブロックである。図５（Ｂ）に示すフリーリスト４０では，バッファブロック情報４２ｂ，４２ｄ，…４２ｘを持つブロックがダーティブロックである。他のブロックは，割り当て可能ブロックであり，この割り当て可能ブロックの数が，パーティション毎に設定された割り当て可能ブロック下限値より小さくなると，ダーティブロックのディスク装置３２への書き込みが行われ，ダーティフラグがオフにされる。これにより，図５（Ｃ）に示すように，割り当て可能ブロックの数が，割り当て可能ブロック下限値以上になるように管理される。

【００３６】図６は，本発明の実施例におけるスワップ中のブロックの状態遷移を示す図である。図６（Ａ）に示す状態で，スワップアウトのためにバッファブロックが獲得されると，スワップファイル用ブロックが必要個数確保され，スワップアウトの後，ダーティフラグをオンにして解放される。フリーリスト４０につながれるバッファブロック情報４２の状態は，図６（Ｂ）に示すようになる。スワップインが行われると，スワップファイルとして用いられたブロックのダーティフラグがオフに戻され，図６（Ｃ）に示すような状態に遷移する。

【００３７】図７は，本発明の実施例によるバッファ領域初期化処理フローチャートである。図２に示すバッファ領域初期化処理部１５は，システムの初期化時にファイル管理処理部３１から呼び出される。引数の入力パラメータは，バッファ領域の情報，ブロックサイズ，割り当て可能ブロック下限値である（図７のステップ７０）。バッファ領域初期化処理部１５は，まずステップ７１により，対象領域を指定されたブロックサイズ単位で分割する。次にステップ７２により，各ブロックについて，図４（Ｂ）に示す内容のバッファブロック情報４２を初期化し，フリーリスト４０に入れる。ステップ７３では，指定された割り当て可能ブロック下限値を，パーティション管理情報として設定する。ステップ７４では，全ブロック数を割り当て可能ブロック数として初期化する。ステップ７５では，パーティションＩＤを割り当て，そのパーティションＩＤを呼び出し元のファイル管理処理部３１に返却する（ステップ７６）。

【００３８】ファイル管理処理部３１は，バッファ領域の用途別にバッファ領域初期化処理部１５を呼び出すことにより，ファイルシステム１１で使用するバッファ領域１８を複数個に分割し，用途別にパーティションを設定することができる。

【００３９】図８は，本発明の実施例によるバッファブロック獲得処理フローチャートである。図２に示すバッ

ファブロック獲得処理部１６は，各プログラム３０Ａ～３０Ｂのデータ入出力時にファイル管理処理部３１から呼び出される。入力パラメータは，ブロックの獲得対象となるパーティションを指定するパーティションＩＤ，データ入出力対象のファイルＩＤ，ファイル内位置の情報である（図８のステップ８００）。

【００４０】バッファブロック獲得処理部１６は，まずステップ８０１により，指定されたファイルＩＤとファイル内位置をキーとしたハッシュ値を計算する。次に，計算したハッシュ値をもとに，ステップ８０２により，図４に示す該当パーティションにおけるハッシュテーブル４１のエントリを求め，ハッシュキューを探索する。

【００４１】ステップ８０３の判定により，ファイルＩＤおよびファイル内位置が一致する該当バッファブロック情報４２が見つかったならば，ステップ８１０へ進み，見つからなかった場合には，次のステップ８０４へ進む。

【００４２】該当バッファブロック情報４２が見つからなかった場合，ステップ８０４により，フリーリスト４０を探索し，最初に見つかった割り当て可能ブロックのバッファブロック情報４２をフリーリスト４０から外す。ステップ８０５では，古いハッシュキューからこのブロックのバッファブロック情報４２を外し，ファイルＩＤおよびファイル内位置をキーとして計算したハッシュ値に対応する新しいハッシュキューへ入れる。次のステップ８０６で，ファイルＩＤの設定，ファイル内位置の設定，データ有効フラグのオフなどのバッファブロック情報４２の初期化を行い，ステップ８０７で割り当て可能ブロック数を減らす。

【００４３】ステップ８０８により，割り当て可能ブロック数が割り当て可能ブロック下限値より小さくなったかどうかを判定し，小さくなければステップ８１３へ進む。割り当て可能ブロック下限値より小さければ，ステップ８０９により，フリーリスト４０を探索し，ダーティフラグがオンになっているブロックの内容をディスクへ書き戻し，そのブロックを割り当て可能状態にして，割り当て可能ブロック数を増やす。その後，ステップ８１３へ進む。

【００４４】ステップ８０３の判定で，該当バッファブロック情報４２が見つかった場合，ステップ８１０により，そのバッファブロック情報４２がフリーリスト４０中かどうかを判定する。フリーリスト４０中であれば，ステップ８１１によりそのバッファブロック情報４２をフリーリスト４０から外す。

【００４５】次に，ステップ８１２により，そのバッファブロック情報４２におけるデータ有効フラグをオンにして，ステップ８１３へ進む。ステップ８１３では，獲得したブロックのバッファブロック情報４２を出力情報として，呼び出し元のファイル管理処理部３１へ復帰する。

【００４６】図９は，本発明の実施例によるバッファブロック解放処理フローチャートである。図２に示すバッファブロック解放処理部１７は，ファイル管理処理部３１から呼び出され，獲得したバッファブロックの解放処理を行う。入力パラメータは，解放するブロックの存在するパーティションを指定するパーティションＩＤ，バッファブロック情報，スワップインフラグである（図９のステップ９０）。スワップインフラグは，スワップインが行われたバッファブロックを解放する場合にオン，その他の場合にはオフに設定されるフラグである。

【００４７】バッファブロック解放処理部１７は，まずステップ９１により，該当パーティションのフリーリスト４０に，解放するブロックのバッファブロック情報４２を入れる。次にステップ９２により，スワップインフラグがオンかオフかを判定し，スワップインフラグがオンの場合，すなわちスワップイン時のブロック解放の場合には，ステップ９３によってダーティフラグをオフにする。

【００４８】ステップ９４では，ダーティフラグのオン／オフを判定し，ダーティフラグがオフであれば割り当て可能ブロック数を増やす。その後，ステップ９６で呼び出し元のファイル管理処理部３１へ復帰する。

【００４９】図１０は，本発明の実施例によるスワップ処理フローチャートである。図２に示すファイル管理処理部３１または図示省略したスワップ制御の処理手段によって，図１０（Ａ）に示すスワップアウト処理およば図１０（Ｂ）に示すスワップイン処理によるジョブのスワップ制御が行われる。

【００５０】スワップアウトでは，まずステップ１０１により，システムデータ用バッファ領域１８５のパーティションＩＤと，スワップファイルのファイルＩＤと，ファイル内位置とを指定して，バッファブロック獲得処理部１６を呼び出し，スワップファイル用ブロックを獲得する。次に，ステップ１０２により，獲得したバッファブロックにスワップアウトするジョブのコンテキスト５０を書き込み，ステップ１０３によって，そのブロックのバッファブロック情報４２におけるダーティフラグをオンにする。

【００５１】次に，ステップ１０４により，スワップインフラグをオフにし，ステップ１０５により，パーティションＩＤとバッファブロック情報とスワップインフラグとを引数として，バッファブロック解放処理部１７を呼び出す。バッファブロック解放処理部１７から制御が戻ったならば，ステップ１０６により，全コンテキスト５０のスワップアウトが終了したかどうかを判定し，未処理部分があれば，ステップ１０１へ戻って，次のブロックについて同様に処理を繰り返す。全コンテキスト５０について終了したならば，スワップアウトの処理を終了する。

【００５２】スワップインでは，まずステップ１１１に

より，システムデータ用バッファ領域１８Ｓのパーティション ＩＤと，スワップファイルのファイルＩＤと，スワップイン対象のファイル内位置とを指定して，バッファブロック獲得処理部１６を呼び出す。バッファブロックを獲得したならば，ステップ１１２により，そのバッファブロック情報４２を参照し，データ有効フラグのオン／オフを判定する。データ有効フラグがオンであれば，ステップ１１４へ進む。

【００５３】データ有効フラグがオフであれば，ステップ１１３により，獲得したブロックにディスク上のスワップファイルから該当コンテキスト５０を読み出す。ステップ１１４では，バッファブロックの内容をジョブ領域へ複写し，コンテキスト５０の復元を行う。その後，ステップ１１５により，スワップインフラグをオンにし，ステップ１１６により，パーティションＩＤとバッファブロック情報とスワップインフラグとを引数として，バッファブロック解放処理部１７を呼び出す。バッファブロック解放処理部１７から制御が戻ったならば，ステップ１１７により，全コンテキスト５０のスワップインが終了したかどうかを判定し，未処理部分があれば，ステップ１１１へ戻って，次のブロックについて同様に処理を繰り返す。全コンテキスト５０について終了したならば，スワップインの処理を終了する。

【００５４】

【発明の効果】以上説明したように，本発明によれば，各入出力の用途に適合した用途別のバッファ管理を行うことが可能となり，システム全体の性能向上に寄与するところが大きい。特に，例えば数十台～数千台というような多数の処理装置から構成される並列計算機システム等においては，各処理装置において限られたバッファ領域を効率よく管理し利用する技術が必要とされ，本発明によれば，このようなシステムにおいてバッファを利用した入出力処理の高速化と，バッファ領域の有効利用を実現することが可能になる。

【図面の簡単な説明】

【図１】本発明の構成例を示す図である。

【図２】本発明の実施例におけるファイルシステムの構成例説明図である。

【図３】本発明の実施例におけるバッファ領域の分割説明図である。

【図４】本発明の実施例におけるバッファブロックの管理情報説明図である。

【図５】本発明の実施例におけるスワップ制御説明図である。

【図６】本発明の実施例におけるスワップ中のブロックの状態遷移を示す図である。

【図７】本発明の実施例によるバッファ領域初期化処理フローチャートである。

【図８】本発明の実施例によるバッファブロック獲得処理フローチャートである。

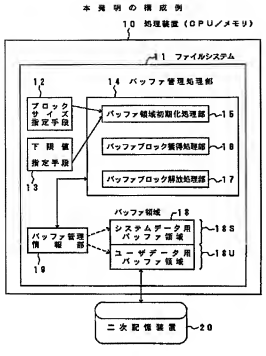
【図９】本発明の実施例によるバッファブロック解放処理フローチャートである。

【図１０】本発明の実施例によるスワップ処理フローチャートである。

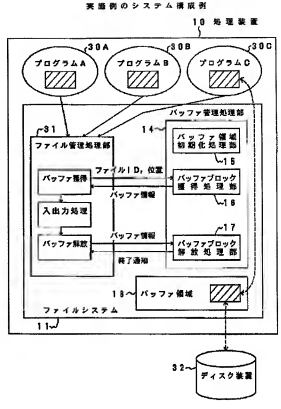
【符号の説明】

１０　処理装置
１１　ファイルシステム
１２　ブロックサイズ指定手段
１３　下限値指定手段
１４　バッファ管理処理部
１５　バッファ領域初期化処理部
１６　バッファブロック獲得処理部
１７　バッファブロック解放処理部
１８　バッファ領域
１９　バッファ管理情報部
２０　二次記憶装置

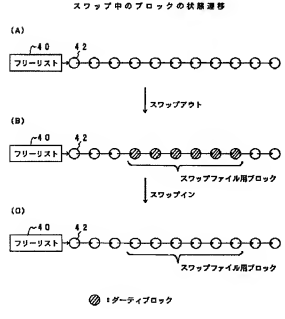
【図1】


本発明の構成例
10 処理装置（CPU／メモリ）
11 ファイルシステム
12
ブロックサイズ指定手段
13
下限値指定手段
14 バッファ管理処理部
バッファ領域初期化処理部 15
バッファブロック獲得処理部 16
バッファブロック解放処理部 17
バッファ領域 18
システムデータ用バッファ領域 18S
ユーザデータ用バッファ領域 18U
バッファ管理情報部 19
二次記憶装置 20

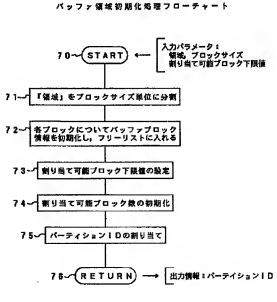
【図2】


実施例のシステム構成例
10 処理装置
プログラムA 30A
プログラムB 30B
プログラムC 30C
バッファ管理処理部 14
ファイル管理処理部 31
バッファ領域初期化処理部 15
バッファ獲得
ファイルID，位置
バッファブロック獲得処理部 16
バッファ情報
入出力処理
バッファ解放
バッファ情報
バッファブロック解放処理部 17
終了通知
バッファ領域 18
ファイルシステム 11
ディスク装置 32

【図6】

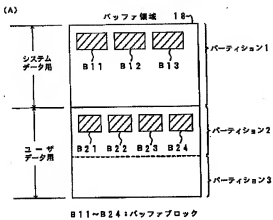

スワップ中のブロックの状態遷移
(A)
40 42
フリーリスト
スワップアウト
(B)
フリーリスト
スワップファイル用ブロック
スワップイン
(C)
フリーリスト
スワップファイル用ブロック
：ダーティブロック

【図7】


バッファ領域初期化処理フローチャート
70 START
入力パラメータ：領域，ブロックサイズ，割り当て可能ブロック下限値
71 「領域」をブロックサイズ単位に分割
72 各ブロックについてバッファブロック情報を初期化し，フリーリストに入れる
73 割り当て可能ブロック下限値の設定
74 割り当て可能ブロック数の初期化
75 パーティションIDの割り当て
76 RETURN
出力情報：パーティションID

【図3】

バッファ領域の分割説明図

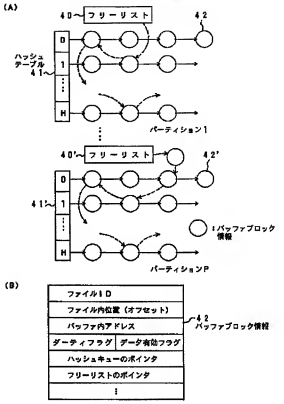


(B)　パーティション管理情報

パーティション番号	ブロックサイズ	割り当て可能ブロック下限値
1	S1	L1
2	S2	L2
3	S3	L3

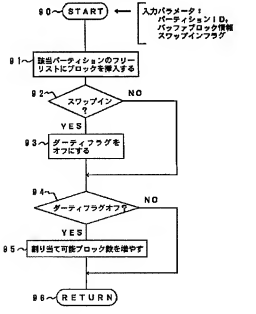
【図4】

バッファブロックの管理情報説明図

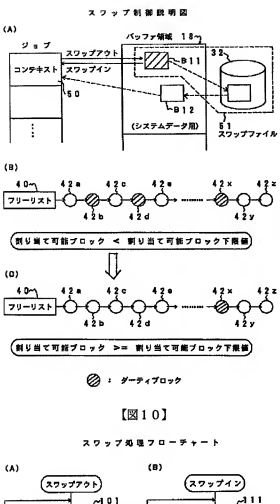


【図9】

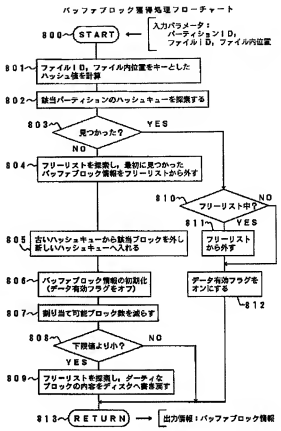
バッファブロック解放処理フローチャート



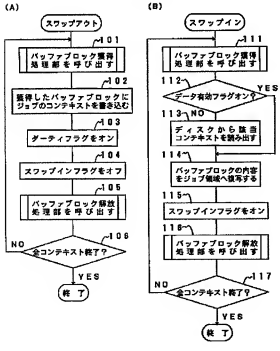
【図5】


スワップ制御説明図
(A)
ジョブ
スワップアウト
コンテキスト
スワップイン
50
バッファ領域 18
B11
32
B12
(システムデータ用)
51
スワップファイル
(B)
40
フリーリスト
42a
42c
42e
42x
42z
42b
42d
42y
割り当て可能ブロック ＜ 割り当て可能ブロック下限値
(C)
割り当て可能ブロック ＞＝ 割り当て可能ブロック下限値
：ダーティブロック

【図8】


バッファブロック獲得処理フローチャート
800 START
入力パラメータ：
パーティションID，
ファイルID，ファイル内位置
801 ファイルID，ファイル内位置をキーとした
ハッシュ値を計算
802 該当パーティションのハッシュキューを探索する
803 見つかった？
YES
NO
804 フリーリストを探索し，最初に見つかった
バッファブロック情報をフリーリストから外す
810 フリーリスト中？
NO
811 YES
フリーリスト
から外す
805 古いハッシュキューから該当ブロックを外し
新しいハッシュキューへ入れる
806 バッファブロック情報の初期化
(データ有効フラグをオフ)
データ有効フラグを
オンにする
812
807 割り当て可能ブロック数を減らす
808 下限値より小？
NO
YES
809 フリーリストを探索し，ダーティな
ブロックの内容をディスクへ書き出す
813 RETURN
出力情報：バッファブロック情報

【図10】


スワップ処理フローチャート
(A)
スワップアウト
101
バッファブロック獲得
処理部を呼び出す
102
獲得したバッファブロックに
ジョブのコンテキストを書き込む
103
ダーティフラグをオン
104
スワップインフラグをオフ
105
バッファブロック解放
処理部を呼び出す
106
NO
全コンテキスト終了？
YES
終了
(B)
スワップイン
111
バッファブロック獲得
処理部を呼び出す
112
データ有効フラグオン？
YES
113
NO
ディスクから該当
コンテキストを読み出す
114
バッファブロックの内容
をジョブ領域へ復写する
115
スワップインフラグをオン
116
バッファブロック解放
処理部を呼び出す
117
NO
全コンテキスト終了？
YES
終了